View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Limerick Institutional Repository
DOI: http://dx.doi.org/10.1145/2600821.2600839

Artifact-Based Software Process Improvement and Management: A Method Proposal

Marco Kuhrmann
Technische Universität München
Faculty of Informatics
Munich, Germany
kuhrmann@in.tum.de

Sarah Beecham
The Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
sarah.beecham@lero.ie

ABSTRACT

When it comes to software process improvement (SPI), process engineers look for SPI methods to support process analysis, design, realization, deployment, and management. Although a number of different SPI methods and models exist, process engineers tend to view these as too generic, too large, or a poor fit for the organization in which SPI is conducted. A strategy to overcome these shortcomings is to concentrate on the artifacts, which precisely define the desired outcomes, rather than on specific methods. In this paper, we present the Artifact-based Software Process Improvement & Management (ArSPI) model that provides a unified perspective on SPI and company-wide software process management (SPM), the required key artifacts, and the life cycle models. ArSPI is shown to be of practical support to industry who called for a practical way to define the interfaces between SPI projects. This paper concludes with an example of how ArSPI paved the way for several organizations through applying the model in real-world SPI-projects.

Categories and Subject Descriptors

D.2.9 [**Software Engineering Management**]: Software process models

General Terms

Management, Experimentation

Keywords

software process improvement, software process management, SPI, SPM, artifact-orientation, method proposal

1. INTRODUCTION

Software process improvement (SPI) is recognized as an important success factor for companies operating in a competitive market [8], as improved processes can lead to increased speed of development, product quality, and product reliability. A number of standardized SPI approaches exist to support process engineers in managing SPI, e.g., reference models such as CMMI [4] or ISO 15504 [9], as well as more specific and light-weight SPI models, e.g., LAPPI [14], or BG-SPI [2]. However, process engineers and process consumers often complain about approaches that are too generic, too comprehensive, or that are inappropriate for the actual project or company context [15]. Also, SPI is costly and improved processes need time to be disseminated, making the impact of SPI activities hard to measure. Therefore, project managers are often reluctant to implement SPI, and standard approaches are often neglected [3, 5, 15].

The challenges inherent in SPI projects can to a certain extent be alleviated by appointing an experienced process engineer who will play a key role in the SPI project. Furthermore, adopting an *artifact-based* approach in which analysis/design concepts and the desired outputs are viewed as 'artifacts' allows for a seamless SPI cycle (analyze, design, realize, deploy, improve). Artifacts materialize as tangible units at different levels of abstraction (Sect. 2). Experiences gathered during the development and continuous improvement of the V-Modell XT (the German standard software development process, [16]) and lab-based investigations showed taking an artifact approach can benefit process engineers in several ways. For example, the consistent terminology used in artifact modeling helps to avoid misunderstandings [11], artifacts support communication and data exchange [13], and artifacts are easier to evaluate in quality assurance than "performed" activities. A focus on artifacts gives process engineers the freedom to use those methods for artifact creation best serving the actual project context, e.g., creating text documents, or creating comprehensive models.

Contribution. In this paper, we propose a model for an *Artifact-based Software Process Improvement & Management* (ArSPI). The presented model emerges from a number of SPI projects conducted in Germany and Eastern Europe. It puts an emphasis on the artifacts produced in SPI projects, and it also defines interfaces between SPI projects and provides company-wide SPM through the use of artifacts. We introduce the ArSPI model, describe its components, and give examples of how the model is applied in practice.

Outline. The paper is organized as follows. In Sect. 2, we briefly introduce the terminology, the background, and the most relevant related work. In Sect. 3, we introduce the ArSPI model, its components, and show the application in SPI projects as well as on the organization level. We conclude the paper in Sect. 4.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICSSP '14, May 26–28, 2014, Nanjing, China
Copyright 2014 ACM 978-1-4503-2754-1/14/05 ...$15.00.

© ACM. PREPRINT. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution.
The definitive version was published in the conference/workshop proceedings.
DOI: http://dx.doi.org/10.1145/2600821.2600839

2. RELATED WORK

We briefly introduce the terminology used, discuss some background information, and the most relevant related work in the context of our method proposal.

Background & Terminology. The term *artifact* and its associated meaning is central to our study. An artifact according to [6] is defined as any (tentative) deliverable that is created, consumed, or modified by an activity. Artifacts have a type, define a structure, and are embedded in a dependency model. A set of artifacts and their dependencies form an *artifact model*. According to the level of abstraction and formalization, we distinguish between analysis, design, realization, and project artifacts. Analysis artifacts, usually, provide an informal description of an investigated concept. A design artifact is used to develop a model (e.g., a role model) that still does not need full formalization. A realization artifact is a formal unit, which is part of a process realization, e.g., a process asset realized in a process framework, and that potentially allows for process enactment. Finally, *project artifacts* are instances in (SPI) projects.

Related Work. Due to space limitations, we can only scratch the surface of related work. The proposed model addresses SPI as well as SPM, and places an emphasis on the organization of (long-term) SPI programs. CMMI or ISO 15504 are so-called normative models that focus on assessment and maturity determination. They are generic and non-prescriptive leaving the process engineer to tailor them to the needs of their own organization. Therefore, these models do not support the organization of SPI projects nor the implementation of improvements. Discussion on their strengths and weaknesses are the subject of much research, e.g., from [3, 5, 15]. Light-weight SPI models, such as PROCESSUS [8], LAPPI [14], COMPETISOFT [7], or BG-SPI [2] usually address small-to-medium sized companies and provide detailed guideline regarding the approach to conduct SPI. However, if these models define artifacts they usually just name the artifacts and give examples, but omit detailed structure and dependencies. With the proposed model, we shift the focus to the artifacts in terms of the objects that are created and exchanged in SPI projects. We therefore provide a much needed context for the artifacts forming a central part of all SPI projects.

3. THE ARSPI MODEL

We introduce the *Artifact-based Software Process Improvement & Management* (ArSPI) model. We give an overview in Sect. 3.1, followed by a description of the ArSPI artifact model (Sect. 3.2), and the life cycle and organization model (Sect. 3.3). Finally, we give two examples (based on project experience) to illustrate the use of ArSPI: We first show how SPI projects can be conducted using ArSPI (Sect. 3.4) and, second, show how ArSPI can be used to install a long-term SPI strategy at organization level in Sect. 3.5.

3.1 Introduction to ArSPI

The ArSPI[1] model is an artifact-based approach to organize SPI and, thus, focuses on the artifacts being produced in SPI projects—it focuses on the "what". Therefore, ArSPI defines a comprehensive, but flexible artifact model that addresses SPI projects as well as a company-wide SPM. ArSPI consists of three parts (Fig. 1).

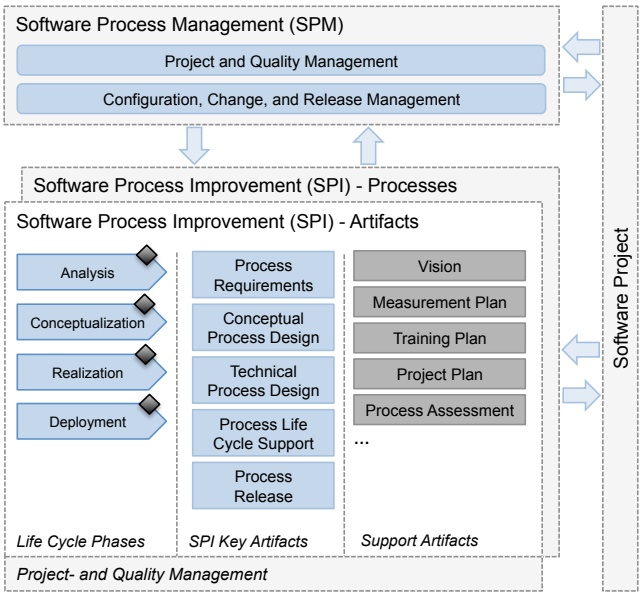

Figure 1: The ArSPI model (overview).

SPI Projects. ArSPI characterizes an SPI project by defining 5 mandatory *key artifacts* to be created (Table 1), a set of 24 complementing *support artifacts*, *life cycle phases* to which the artifacts are assigned for the project organization, and a (rudimentary) *process model*.

Company-wide SPM. A company-wide software process management (SPM) implements a long-term SPI strategy, owns the software processes, initiates particular improvement endeavors, and deploys process releases for the process consumers. For this, ArSPI defines artifacts and processes to (1) define interfaces between the company-wide SPM and particular SPI projects, and (2) to establish the necessary management and administration processes.

Software Projects. The primary audience of SPI are software projects that consume (improved) processes, and that produce experiences and data for further improvements.

3.2 ArSPI Artifact Model

In this section, we describe the artifact model of ArSPI in more detail. We define the key artifacts, briefly describe how the artifacts are modeled and how particular artifacts materialize in projects.

ArSPI defines 5 key artifacts (Table 1), which have to be created in every SPI project. Basically, ArSPI proposes a two-staged design process (reflected by the artifacts CPD and TPD) to separate conceptualization and (technical) realization. However, in view of the fact that SPI projects can be performed on a small scale, CPD and TPD can be integrated into a unified *Process Design* artifact (Sect. 3.4). The key artifacts are assigned to the SPI project life cycle phases (Fig. 1), namely: PRQ $\mapsto$ Analysis, CPD $\mapsto$ Conceptualization, TPD $\mapsto$ Realization, and PR $\mapsto$ Deployment. The PLC artifact can be created early in the cycle in the Analysis phase, however, it must be created by the time the

[1]Detailed information on the models and processes can be depicted from our complementing technical report [10].

© ACM. PREPRINT. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution.
The definitive version was published in the conference/workshop proceedings.
DOI: http://dx.doi.org/10.1145/2600821.2600839

Table 1: ArSPI key artifacts.

Id	Artifact Description
PRQ	The *Process Requirements* contain all requirements regarding the process development.
CPD	The *Conceptual Process Design* contains all designs of a process without paying attention to any technical realization. It refines all process-related requirements and transfers them into concrete processes and process parts.
TPD	The *Technical Process Design* refines the conceptual process design regarding a concrete technical implementation and the tool/tool infrastructure to be used for the process's realization.
PLC	The *Process Life Cycle Support* comprehends all information, agreements, and definition regarding complementing processes that support the deployment, training, and further development of a concrete process. The life cycle support comprehends at least agreements for: training, deployment and further development, measurement and evaluation, and change management.
PR	A *Process Release* is a concrete process package that is shipped and deployed.

Deployment phase is reached. The artifact model as such is designed as a comprehensive UML-model to serve several realization options. In the simple case, artifacts (represented as classes) aggregate further fine-grained artifacts. This aggregation structure can be easily transformed into a document structure, e.g., Word. Likewise, the artifact model can also be instrumented in tools (e.g., for design and enactment [12]). Depending on the actual context and the used (project) infrastructure, ArSPI artifacts thus can materialize as documents or computable data of corresponding tools, e.g., presentation slides, Wiki pages, or tickets in a tracking system.

3.3 ArSPI Life Cycle and Organization Model

We briefly describe the life cycle model of SPI projects and the overall organization model. We show, how SPI and SPM are integrated by ArSPI providing a unified view.

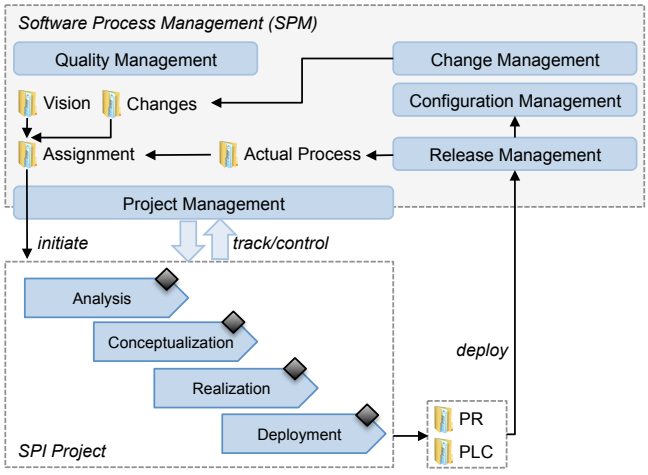

Figure 2: ArSPI organization model (simplified).

Figure 2 presents a simplified view of the life cycle and organization model of ArSPI, and how the quality management process interfaces with project management (an example is shown in Fig. 3). A SPI project starts with a *Project Assignment* (e.g., a contract) from the process-owning company, and is iteratively performed by the process engineers (whereby each iteration comprises up to four phases that are based on the Plan-Do-Check-Act cycle). The goal is to deploy one PR per iteration. PR and PLC are shipped to the organization that includes the PR in the *release management* (usually combined with a *configuration management*), and, eventually, publishes a PR as a new *Actual Process* for use in software projects. A *change management* is enabled for the new PR, and, together with a *quality management*, collects issues and required *Changes* for further improvement cycles. The company-wide *quality management* should also have a *Vision* representing the overall improvement goals, e.g., a certain CMMI level. A *Vision*, a set of *Changes*, and an *Actual Process* as reference are the basic inputs to initiate improvement cycles.

3.4 Example: Performing an SPI Project

We provide an example to illustrate how ArSPI is applied in an SPI project. In Fig. 3, we show the structure of the first two iterations of an SPI project conducted in Eastern Europe (in 2012/2013).

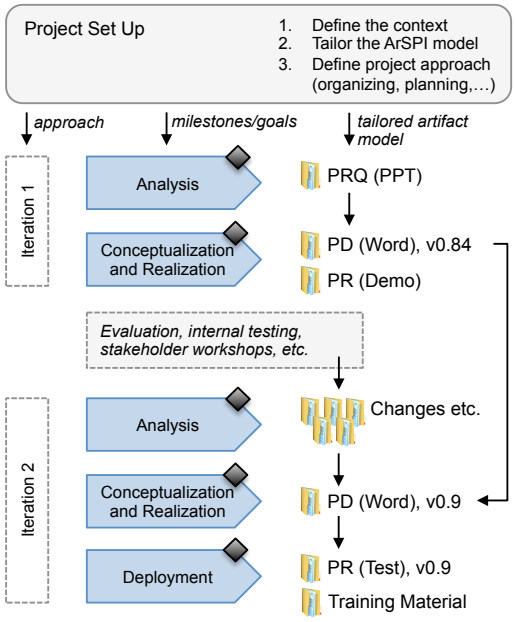

Figure 3: Example SPI project structure.

ArSPI provides SPI projects with structure, and defines results that have to be created, in order to allow for project tracking and progress determination. In the following, we show how to set up a SPI project using ArSPI and give a brief overview of the first two iterations of the SPI project.

SPI Project Set Up. In the first step of the set up, few experience-based questions (e.g., for the context, preknowledge, or deployment and training strategies; [1, 10]) need to be answered to prepare the tailoring of the artifact model. The tailoring is done in the second step (at the

© ACM. PREPRINT. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution.
The definitive version was published in the conference/workshop proceedings.
DOI: http://dx.doi.org/10.1145/2600821.2600839

moment) using a simple checklist. In this step, artifacts to be (not) created are determined, simplification and merging of artifacts is defined, and the materialization of the artifacts is defined, e.g., a Powerpoint presentation for the PRQ. For instance, in Fig. 3, the abbreviation PD means an artifact *Process Design*, which is a merged and simplified artifact that integrates the CPD and TPD artifacts to address smaller SPI projects. Finally, in the third set up step, the overall project approach is defined. This step includes the creation of the project-related manuals, e.g., for project management or quality management procedures.

Iteration 1. Having set up the SPI project, the first SPI artifacts are created. In Fig. 3, the first artifact to be created is the PRQ. The ArSPI model defines several options to provide input for the PRQ's creation, e.g., an actual process (to be improved), assessments, or a vision. After the first analyses, the PD and the PR (as demonstrator) are created. The figure shows the first iteration to be shortened, as it does not contain a deployment phase. In fact, in the project, the selected approach contained an explicitly defined prototyping work package in which the basic requirements should be analyzed and prototypically implemented. For this, only a demonstrator should be delivered as PR, which was then evaluated by the process owners.

Iteration 2. In the second iteration, the PD was refined in response to the client's evaluation and the respective change requests. In Fig. 3, the evolution of the PD is shown. Furthermore, the second iteration should produce a "full" PR for initial deployment and validation purposes. At the same time, the personnel's training was initiated.

Artifact Materialization. In Fig. 3, we only show a few key artifacts required by ArSPI. ArSPI defines the contents of the model, leaving the format open. For example, designs for roles, processes, tailoring and so forth form part of the process design (PD). However, the particular methods for creating the contents are also left open and can thus used according the actual needs, e.g., text-based descriptions, or model-based designs.

3.5 Example: SPI at Organization Level

A special feature of ArSPI is the integration of SPI and SPM. In this section, we show by example how such an integration can be installed, and how organizational (internal) and external triggers affect SPI projects.

In Fig. 4, we provide an example of how ArSPI was implemented in the organization-wide software process management of a German government agency. In this agency, the V-Modell XT was adopted to implement the contracting and software development processes. That is, the process is part of the V-Modell XT process line. The concrete agency's process variant is built on the "V-Modell XT Bund", which is itself a variant of the general "V-Modell XT Reference Model". This small setting shows the demand for a mature process management, as the evolution of the software process depends on internal triggers as well as external ones.

Internal Evolution. In the internally triggered evolution of the software process, the owning agency has their own feedback and improvement cycles. Software developers report problems or propose improvements. The portfolio management and quality management units that own the software process bundle change requests and (new) requirements

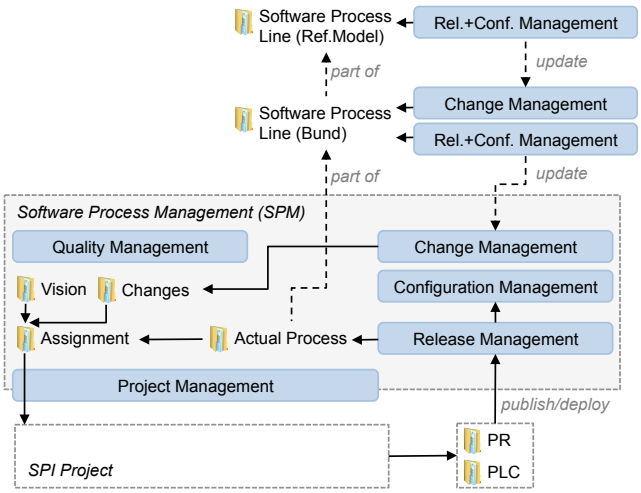

Figure 4: ArSPI in company-wide SPM.

to direct another iteration in the improvement program. Finally, an SPI project is started (Sect. 3.4). At the moment, this agency deploys one PR per year.

Externally Triggered Evolution. While the agency is in full control of their own process variant and directs the improvement, the process variant as such is based on an externally managed reference process (the "V-Modell XT Bund"). This reference process has its own life cycle in which the process is improved. A new "V-Modell XT Bund" PR thus causes an *update trigger* that generates a change request in the agency's change management system. In the next iteration, the externally caused change requests thus becomes an improvement requirement, too. The ArSPI model provides the agency with appropriate information of how the process variant was derived from the reference process (e.g., in the design artifacts, in the *SPLDeltaReport* artifact, etc.) and, thus, helps to determine the changes of the reference process and how these changes affect the process variant (e.g., changed *ProcessAsset*s and, because of the dependency model, transitively affected ones). Due to the fine-grained artifact model, affected artifacts can be identified, and respective work packages to adopt changes can be defined.

As Fig. 4 also shows, the "V-Modell XT Bund" itself is a variant of the "V-Modell XT Reference Model" and, thus, has the same situation of internally and externally triggered evolution. A detailed description of management processes for such settings can be depicted from [10].

ArSPI Implementation. As artifacts basically describe the expected information and structure, tools can be used to represent artifacts in projects, e.g., a *Change* artifact can be represented by a ticket in a tracking system. The use of tools thus helps to reduce the "paper work" in SPI projects [12]. For instance, to support change-, release-, and some aspects of project management (planning, work package creation, etc.), in the referred project, we set-up an IceScrum instance.

4. CONCLUSION & FUTURE WORK

In this paper, we proposed a model for an *Artifact-based Software Process Improvement & Management* (ArSPI). The

© ACM. PREPRINT. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in the conference/workshop proceedings.
DOI: http://dx.doi.org/10.1145/2600821.2600839

presented model emerges from a number of SPI projects conducted in Germany and Eastern Europe. It puts an emphasis on the artifacts produced in SPI projects, and it also defines interfaces between SPI projects and provides company-wide SPM through the use of artifacts.

Impact/Implications. With ArSPI we do not neglect any of the established SPI approaches (e.g., Sect. 2). Moreover, we propose an experience-based framework to set up and organize SPI. Using a precisely defined artifact model, we support process engineers in setting up SPI projects, defining work packages, and providing a means to establish measurement and project tracking. Furthermore, the ArSPI model aims to capture all artifacts being produced in SPI projects and relate them to each other. Therefore, ArSPI also support quality assurance and consistency checks of SPI projects. For example, ArSPI can help to answer questions like "What is the realization for an identified concept?" ArSPI is highly flexible and supports smaller SPI projects (Sect. 3.4) as well as a company-wide SPM (Sect. 3.5).

Future Work. ArSPI is currently published as a 0.9 release, which was crafted from a series of SPI projects, initially validated in an academic context, and piloted in industry. The initial validation and application in practice improved our knowledge and showed opportunities for improvement that we now improve through an iterative implementation and validation cycle. Special emphasis is placed on the refinement of the artifact model, improvement of the tailoring mechanisms, and extension of the evaluation model. Furthermore, although ArSPI is basically designed as a method-agnostic approach, we started to collect best practices, and to provide specific guidance (methods, templates, measurement instruments, etc.) to support process engineers.

Finally, we cordially invite other researchers to use ArSPI and to help to improve the artifact-based SPI approach.

5. ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (`www.lero.ie`).

6. REFERENCES

[1] O. Armbrust, J. Ebell, U. Hammerschall, J. Münch, and D. Thoma. Experiences and results from tailoring and deploying a large process standard in a company. *Software Process: Improvement and Practice*, 13(4):301–309, July 2008.

[2] B. Aysolmaz and O. Demirörs. A detailed software process improvement methodology: BG-SPI. In *European System & Software Process Improvement and Innovation (EuroSPI)*, Communications in Computer and Information Science, pages 97–108. Springer, 2011.

[3] J. L. Boria. A framework for understanding software process improvement's return on investment. In *Portland International Conference on Management and Technology (PICMET)*, pages 847–851. IEEE, 1997.

[4] CMMI Product Team. CMMI for Development, Version 1.3. Technical Report CMU/SEI-2010-TR-033, Software Engineering Institute, CMU, 2010.

[5] G. Coleman and R. O'Connor. Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, 81(5):772–784, 2008.

[6] D. M. Fernández, B. Penzenstadler, M. Kuhrmann, and M. Broy. A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering. In *13th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Lecture Notes in Computer Science, pages 183–197. Springer, 2010.

[7] F. García, M. Piattini, F. Ruiz, F. J. Pino, and C. Alquicira. Software process improvement: The competisoft project. *Computer*, 40(10):21–28, 2007.

[8] R. V. Horvat, I. Rozman, and J. Györkös. Managing the complexity of spi in small companies. *Software Process: Improvement and Practice*, 5(1):45–54, 2000.

[9] ISO. Software Process Assessment - Part 4: Guidance on use for process improvement and process capability determination. Technical Report ISO/IEC 15504-4:2004, International Organization for Standardization, 2004.

[10] M. Kuhrmann. Arspi: An artifact model for software process improvement and management. Research Report TUM-I1337, TU München, 2013.

[11] M. Kuhrmann, D. M. Fernández, and A. Knapp. Who Cares About Software Process Modelling? A First Investigation About the Perceived Value of Process Engineering and Process Consumption. In *International Conference on Product Focused Software Development and Process Improvement (Profes)*, Lecture Notes in Computer Science, pages 138–152. Springer, 2013.

[12] M. Kuhrmann, G. Kalus, and M. Then. The process enactment tool framework – transformation of software process models to prepare enactment. *Science of Computer Programming*, 79(1):172–188, 2014.

[13] M. Kuhrmann, C. Lange, and A. Schnackenburg. A survey on the application of the V-Modell XT in german government agencies. In *Conference on European System & Software Process Improvement and Innovation (EuroSPI)*, Communications in Computer and Information Science, pages 49–60. Springer, 2011.

[14] A. Raninen, J. J. Ahonen, H.-M. Sihvonen, P. Savolainen, and S. Beecham. LAPPI: A light-weight technique to practical process modeling and improvement target identification. *Journal of Software: Evolution and Process*, 25(9):915–933, 2012.

[15] M. Staples, M. Niazi, R. Jeffery, A. Abrahams, P. Byatt, and R. Murphy. An exploratory study of why organizations do not adopt CMMI. *Journal of Systems and Software*, 80(6):883–895, 2007.

[16] Weit e.V. The V-Modell XT Online Portal. Online `http://www.v-modell-xt.de/`.